UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/823,155	03/29/2001	Daniel R. Gaur	42390P10774	8114

8791 7590 06/02/2004

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD, SEVENTH FLOOR
LOS ANGELES, CA 90025

EXAMINER	
BROSS, EDWARD J	

ART UNIT	PAPER NUMBER
2126	7

DATE MAILED: 06/02/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No	Applicant(s)	
	09/823,155	GAUR ET AL.	
	Examiner	Art Unit	
	Edward Bross	2126	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 March 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-35 are pending in this application.

Claim Rejections - 35 USC § 102

2. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

3. Claims 1-35 are rejected under 35 U.S.C. 102(b) as being clearly anticipated by Linux (version 2.3.99-pre3 March 23, 2000) described in Linux Device Drivers (Rubini and Corbet).

4. As to claim 1, Linux teaches the invention substantially as claimed including a method comprising:

requesting a first deferred procedure call for a first interrupt event associated with a source (e.g. request_irq(unsigned int irq, void (*handler)(int, void *, struct pt_regs *), unsigned long flags, const char *dev_name, void *dev_id) as defined in arch/i386/kernel/irq.c and used in drivers/net/sis900.c where 'handler' is the deferred procedure call which will be requested for a network interrupt);

requesting at least one other deferred procedure call for a second interrupt event associated with the source (e.g. the same handler as above which will be called for a second network interrupt);

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource (e.g. request_irq(...) above "...allocates interrupt resources..." arch/i386/kernel/irq.c line 640);

processing the first interrupt event with the first deferred procedure call (e.g. the deferred procedure will be called when the corresponding interrupts occur arch/i386/kernel/irq.c lines 439-443); and

processing the second interrupt event with at least one other deferred procedure call (e.g. same as above when the second interrupt event occurs after the first interrupt event).

5. As to claim 2, Linux discloses the limitations of claim 1 as above and:

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource comprising a processor exhibiting a single thread of execution (e.g. when running on a single processor system, since as implied in Rubini and Corbet by endnote 38 on page 36, the kernel is not preemptable, there is only a single kernel thread of execution outside of interrupt handlers. If only fast handlers are used, interrupt reporting will be disabled, ensuring that both handlers will run sequentially on the same thread, p.11 "Fast and Slow Handlers"); and

executing the first deferred procedure call and the at least one other deferred procedure call on the single thread (e.g. as above, these procedures will be called on the single thread).

6. As to claim 3, Linux discloses the limitations of claim 1 as above and:

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource comprising a processor exhibiting a plurality of threads (e.g. the top half routine, the bottom half routines, and other user processes running on the processor Rubini and Corbet p. 17 "Tasklets and Bottom-Half Processing"); and

executing the first deferred procedure call on one thread of the plurality of threads while executing the at least one other deferred procedure call on another thread of the plurality of threads (e.g. when processing an interrupt in a bottom half routine and a second interrupt occurs, processing of the second interrupt will be done by a top half routine in a separate thread from the currently executing bottom half, Rubini and Corbet p 17 "Tasklets and Bottom-Half Processing").

7. As to claim 4, Linux discloses the limitations of claim 1 as above and:

assigning the first deferred procedure call to a resource comprising a first thread of a processor (e.g. during top half processing of an interrupt, a bottom half is scheduled, Rubini and Corbet p 17 "Tasklets and Bottom-Half Processing");

assigning the at least one other deferred procedure call to a resource comprising a second thread of the processor (e.g. during top half processing of a second interrupt, another bottom half is scheduled, Rubini and Corbet p 17 "Tasklets and Bottom-Half Processing"); and

executing the first deferred procedure call on the first thread while executing the at least

one other deferred procedure call on the second thread. (e.g. as described above, the two handlers will each process their respective interrupts in separate threads)

8. As to claim 5, Linux discloses the limitations of claim 1 as above and:

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource comprising a multi-processor system (e.g. Rubini and Corbet p. 18, third paragraph "SMP systems"); and

executing the first deferred procedure call on one processor of the multi-processor system while executing the at least one other deferred procedure call on another processor of the multi-processor system (e.g. Rubini and Corbet p. 18, third paragraph "Tasklets can run in parallel with other tasklets on SMP systems." where tasklets are a type of bottom half handler).

9. As to claim 6, Linux discloses the limitations of claim 1 as above and:

assigning the first deferred procedure call to a resource comprising a first processor;

assigning the at least one other deferred procedure call to a resource comprising a second processor (e.g. Rubini and Corbet p. 18 "Tasklets are also guaranteed to run on the same CPU as the function that first schedules them"); and

executing the first deferred procedure call on the first processor while executing the at least one other deferred procedure call on the second processor (e.g. as above, when the tasklets are executed one will be on the first processor, the other on the second processor).

10. As to claim 7, Linux discloses the limitations of claim 1 as above and:

processing another interrupt event with the first deferred procedure calls or the at least one other deferred procedure call (e.g. this will occur implicitly whenever the first interrupt occurs again).

11. As to claim 8, Linux discloses a method comprising:

requesting a first deferred procedure call for a first interrupt event associated with a source (e.g. request_irq(unsigned int irq, void (*handler)(int, void *, struct pt_regs *), unsigned long flags, const char *dev_name, void *dev_id) as defined in arch/i386/kernel/irq.c and used in drivers/net/sis900.c where 'handler' is the deferred procedure call which will be requested for a network interrupt);

requesting at least one other deferred procedure call for a second interrupt event associated with the source (e.g. the same handler as above which will be called for a second network interrupt); and

processing the first interrupt event with the first deferred procedure call (e.g. the deferred procedure will be called when the corresponding interrupts occur arch/i386/kernel/irq.c lines 439-443) while processing the second interrupt event with the at least one other deferred procedure call (e.g. same as above when the second interrupt event occurs after the first interrupt event).

12. As to claim 9, Linux discloses the limitations of claim 8 as above and:

executing the first deferred procedure call on a first thread of a processor; and

executing the at least one other deferred procedure call on a second thread of the processor (e.g. as described in the above 102 rejection of claim 4, the two handlers will each process their respective interrupts in separate threads).

13. As to claim 10, Linux discloses the limitations of claim 8 as above and:

executing the first deferred procedure call on a first processor; and

executing the at least one other deferred procedure call on a second processor.

(e.g. Rubini and Corbet p. 18 "Tasklets are also guaranteed to run on the same CPU as the function that first schedules them" implying that when two interrupts are delivered to two different CPUs, the tasklets, or deferred procedures, they will each run on separate processor).

14. As to claim 11, Linux discloses the limitations of claim 8 as above and:

processing another interrupt event with the first deferred procedure calls or the at least one other deferred procedure call (as stated in the above 102 rejection of claim 7, e.g. this will occur implicitly whenever the first interrupt occurs again).

15. As to claim 12, Linux discloses a driver comprising:

an interrupt handler to identify interrupt events associated with a source (e.g. arch/i386/kernel/irq.c lines 427-451

handle_IRQ_event(...)); and

at least two deferred procedure calls, each of the at least two deferred procedure calls to process at least one of the interrupt events (e.g. drivers/net/sis900.c sis900_interrupt(...) lines 850-897 which will be called twice for two network interrupt events).

16. As to claim 13, Linux discloses the limitations of claim 12 as above and:

the interrupt handler to assign the at least two deferred procedure calls to a resource for execution (e.g. drivers/net/sis900.c line 476).

17. As to claim 14, Linux discloses the limitations of claim 12 as above and:

the interrupt handler to assign on of the at least two deferred procedure calls to a first resource for execution and another of the at least two deferred procedure calls to a second resource for execution (e.g. occurs when the two interrupts are handled by two different processors where the deferred procedures will be executed on the different processors. The use of spinlocks in sis900_interrupt(...) implies that these functions may run in parallel on different processors in SMP systems).

18. As to claim 15, Linux discloses a computer system comprising:

a driver stored in a memory of the computer system, the driver including

an interrupt handler to identify interrupt events associated with a source (e.g. arch/i386/kernel/irq.c lines

427-451 handle_IRQ_event(...)); and

at least two deferred procedure calls, each of the at least two deferred procedure calls to process at least one of the interrupt events (e.g. drivers/net/sis900.c sis900_interrupt(...) lines 850-897 which will be called twice to handle two successive network interrupt events), and

a processor to execute the at least two deferred procedure calls (e.g. the implied processor on which the Linux kernel is run).

19. As to claim 16, Linux discloses the limitations of claim 15 as above and:

the interrupt handler to assign the at least two deferred procedure calls to a single thread exhibited by the processor for execution (e.g. when running on a single processor system, since as implied in Rubini and Corbet by endnote 38 on page 36, the kernel is not preemptable, there is only a single kernel thread of execution outside of interrupt handlers. If only fast handlers are used, interrupt reporting will be disabled, ensuring that both handlers will run sequentially on the same thread, p.11 "Fast and Slow Handlers").

20. As to claim 17, Linux discloses the limitations of claim 15 as above and:

the interrupt handler to assign a first of the at least two deferred procedure calls to one thread of the processor and another of the at least two deferred procedure calls to a second thread of the processor for execution (e.g. col. 14 line 65 – col. 15 line 19).

21. As to claim 18, Linux discloses the limitations of claim 15 as above and:

the interrupt handler to assign one of the at least two deferred procedure calls to the processor and another of the at least two deferred procedure calls to a second processor (e.g. Rubini and Corbet p. 18 "Tasklets are also guaranteed to run on the same CPU as the function that first schedules them" implying that when two interrupts are delivered to two different CPUs, the tasklets, or deferred procedures, they will each run on separate processor).

22. As to claim 19, Linux discloses the limitations of claim 15 as above and:

wherein the source comprises a peripheral device coupled with the computer system (e.g. the sis900 network card associated with the sis900_interrupt(...)).

23. As to claim 20, it is rejected for the same reason as claim 1 above.

24. As to claim 21, Linux discloses the limitations of claim 20 as above and:

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource comprising a processor exhibiting a single thread of execution (e.g. when running on a single processor system, since as implied in Rubini and Corbet by endnote 38 on page 36, the kernel is not preemptable, there is only a single kernel thread of execution outside of interrupt handlers. If only fast handlers are used, interrupt reporting will be disabled, ensuring that both handlers will run sequentially on the same thread, p.11 "Fast and Slow Handlers"); and

execute the first deferred procedure call and the at least one other deferred procedure call

on the single thread (e.g. as above, these procedures will be called on the single thread).

25. As to claim 22, Linux discloses the limitations of claim 20 as above and:

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource comprising a processor exhibiting a plurality of threads (e.g. the top half routine, the bottom half routines, and other user processes running on the processor Rubini and Corbet p. 17 "Tasklets and Bottom-Half Processing"); and

execute the first deferred procedure call on one thread of the plurality of threads while executing the at least one other deferred procedure call on another thread of the plurality of threads (e.g. when processing an interrupt in a bottom half routine and a second interrupt occurs, processing of the second interrupt will be done by a top half routine in a separate thread from the currently executing bottom half, Rubini and Corbet p 17 "Tasklets and Bottom-Half Processing").

26. As to claim 23, Linux discloses the limitations of claim 20 as above and:

assigning the first deferred procedure call to a resource comprising a first thread of a processor (e.g. during top half processing of an interrupt, a bottom half is scheduled, Rubini and Corbet p 17 "Tasklets and Bottom-Half Processing");

assigning the at least one other deferred procedure call to a resource comprising a second

thread of the processor (e.g. during top half processing of a second interrupt, another bottom half is scheduled, Rubini and Corbet p 17 "Tasklets and Bottom-Half Processing"); and

executing the first deferred procedure call on the first thread while executing the at least one other deferred procedure call on the second thread. (e.g. as described above, the two handlers will each process their respective interrupts in separate threads)

27. As to claim 24, Linux discloses the limitations of claim 20 as above and:

assigning the first deferred procedure call and the at least one other deferred procedure call to a resource comprising a multi-processor system (e.g. Rubini and Corbet p. 18, third paragraph "SMP systems"); and

executing the first deferred procedure call on one processor of the multi-processor system while executing the at least one other deferred procedure call on another processor of the multi-processor system (e.g. Rubini and Corbet p. 18, third paragraph "Tasklets can run in parallel with other tasklets on SMP systems." where tasklets are a type of bottom half handler).

28. As to claim 25, Linux discloses the limitations of claim 20 as above and:

assigning the first deferred procedure call to a resource comprising a first processor;

assigning the at least one other deferred procedure call to a resource comprising a second processor (e.g. Rubini and Corbet p. 18 "Tasklets are also guaranteed to run on the same CPU as the function that first schedules them"); and

executing the first deferred procedure call on the first processor while executing the at least one other deferred procedure call on the second processor (e.g. as above, when the tasklets are executed one will be on the first processor, the other on the second processor).

29. As to claim 26, Linux discloses the limitations of claim 20 as above and:

processing another interrupt event with the first deferred procedure calls or the at least one other deferred procedure call (e.g. this will occur implicitly whenever the first interrupt occurs again).

30. As to claim 27, it is rejected for the same reason as claim 8 above.

31. As to claim 28, Linux discloses the limitations of claim 27 as above and:

executing the first deferred procedure call on a first thread of a processor; and

executing the at least one other deferred procedure call on a second thread of the processor (e.g. as described in the above 102 rejection of claim 4, the two handlers will each process their respective interrupts in separate threads).

32. As to claim 29, Linux discloses the limitations of claim 27 as above and:

executing the first deferred procedure call on a first processor; and

executing the at least one other deferred procedure call on a second processor.

(e.g. Rubini and Corbet p. 18 "Tasklets are also guaranteed to run on the same CPU as the function that first schedules them" implying that when two interrupts are delivered to two different CPUs, the tasklets, or deferred procedures, they will each run on separate processor).

33. As to claim 30, Linux discloses the limitations of claim 27 as above and:

processing another interrupt event with the first deferred procedure calls or the at least one other deferred procedure call (as stated in the above 102 rejection of claim 7, e.g. this will occur implicitly whenever the first interrupt occurs again).

34. As to claims 31-35, Linux discloses that the source comprises a peripheral device of a computer system (e.g. the sis900 network card associated with the drivers/net/sis900.c driver).

Response to Arguments

35. Applicant's arguments filed March 22, 2004 have been fully considered but they are not persuasive.

36. In the remarks, Applicant argued in substance that (1) Linux does not teach the limitation of a driver having two or more deferred procedure calls that are associated with the same source of interrupts; (2) Rubini is not prior art as to the instant application.

37. Examiner respectfully traverses Applicant's remarks:

A. As to point (1), Linux meets the cited limitations as shown through the mapping provided in the claim rejections above.

B. As to point (2), the features described by Rubini are present in the Linux Kernel Version 2.3.99-pre3 dated March, 2000 which is before the filing date of this application. Rubini does not add additional subject matter to the Linux Kernel but only presents it in a more accessible manner, thus the publication date of Rubini is not relevant as to anticipation of the claimed invention.

38. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

39. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Edward Bross whose telephone number is 305-8754. The examiner can normally be reached on Mon-Fri 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 305-9678. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

EB

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100